SISTEMAS DE INFORMAÇÃO

FUNDAMENTOS DE COMPILADORES

COMPILADOR Cmm

Trabalho apresentado como parte das avaliações parciais da disciplina Fundamentos de Compiladores do curso de Sistemas de Informação do Campus I da UNEB.

ERIC MAIA PEREIRA CARVALHO

2017.2

Introdução

O projeto do compilador foi construído com base nos conhecimentos passados em aula e seguindo as especificações fornecidas pelo professor da disciplina. A primeira etapa envolveu a construção do autômato e do módulo de análise léxica, a segunda envolveu a construção do analisador sintático e gerenciador da tabela de símbolos, e a última etapa, o analisador semântico e gerador de código.

Análise Léxica

Na etapa de construção do analisador léxico, primeiro foi construído um autômato finito determinístico para o reconhecimento dos tokens definidos no documento de especificação da linguagem fornecido.

Foi criado um novo projeto no CodeBlocks e um arquivo header (.h) para o analisador léxico contendo as definições de enumeradores e tamanhos para as tabelas de palavras reservadas, sinais e literais, uma estrutura do tipo struct para guardar as informações relativas aos tokens, além de declarações de funções a serem implementadas.

No arquivo .c foram implementadas as funções de reconhecimento de letras e dígitos, montagem das tabelas, impressão em tela e a função principal do analisador léxico, que instancia uma nova estrutura de token, lê cadeias de caracteres em um arquivo, identifica seus tipos através das condições utilizando um padrão de reconhecimento definido pelo autômato, e retorna a estrutura de token com os dados necessários, ou, em caso de erro léxico encontrado, imprime a mensagem de erro apropriada.

Análise Sintática

Foi criado um novo header para o analisador sintático, contendo enumeradores para as categorias e tipos de identificadores, um novo tipo de estrutura para variáveis, funções e parâmetros e as declarações das funções a serem implementadas.

Para as chamadas de função, utilizou-se as regras de produção da gramática definidas no documento de especificação da linguagem.

A função prog() invoca o analisador léxico duas vezes, uma para capturar um token e outra para capturar o token seguinte (para posterior identificação de parêntese ou ponto-e-vírgula), e chama a função decl(). A função decl() foi removida da especificação mais atual fornecida, mas foi decidido mantê-la por questões de tempo insuficiente para retrabalho e correção dos erros.

A função decl() espera receber um identificador (tipo, semretorno ou prototipo). A função decl() também invoca as funções de reconhecimento de sinais, tipos de parâmetros (com id, ou sem id no caso de prototipo), a presença de abertura e fechamento de parênteses, chaves e colchetes, e reconhecimento de vírgula e ponto-e-vírgula.

A função cmd(), que era invocada pela extinta func() agora é chamada dentro de decl(). cmd() é responsável por tratar estruturas de controle de repetição e condicionais, e atribuições.

atrib() reconhece a atribuição de uma expressão a um identificador.

expr() permite a chamada de uma ou mais expr_simp() e sua comparação com o uso de operadores relacionais, reconhecidos por op_rel().

expr_simp() permite a operação de soma ou subtração ou uma disjunção lógica entre termos. termo() trata as operações de multiplicação e divisão, além da conjunção lógica, garantindo sua precedência sobre as operações anteriores.

fator() reconhece as constantes e chamadas de expressões aninhadas.

Análise Semântica

Não há arquivos dedicados para a análise semântica pois seu funcionamento está atrelado ao processo de análise sintática. Em meio a execução da função decl() na fase da análise sintática, caso não reconheça uma palavra reservada, é apontado erro devido a falta de declaração prévia. Sendo PR, o token é passado para outra função, que primeiro testa se é identificador e então percorre a tabela de símbolos comparando seus elementos ao atual. Em caso de “match”, é exibido erro relativo a redeclaração, senão, é inserido um novo registro na tabela de símbolos, contendo as informações do lexema, seu escopo, categoria e tipo.

Gerenciador de Tabela de Símbolos e Gerenciador de Erros

O gerenciamento da tabela de símbolos foi inserido dentro do tratamento das funções do analisador sintático Foi criado uma estrutura do tipo struct de nome reg_ts para os símbolos, e um array de reg_ts chamado tab_sb, que representa a tabela de símbolos. o programa percorre essa tabela verificando se há redeclaração, e caso não haja, insere novo identificador na tabela.

Os erros são verificados dentro das validações das funções, de acordo com o escopo da função em questão. foi implementada uma função genérica mensagemErro() para apontar a linha do arquivo onde foi encontrado o erro além do tipo de erro encontrado.

Gerador de Código da MP

O gerador de código foi implementado parcialmente seguindo o esquema de tradução fornecido. As funções e variáveis para geração dos labels foram declaradas no arquivo header do analisador sintático. Os comandos de máquina foram inseridos na chamadas apropriadas da árvore conforme sugerido no esquema de tradução.

Conclusão

Os testes realizados com o compilador implementado identificaram alguns erros inesperados sobre os arquivos de código em linguagem Cmm testados, porém não houve tempo hábil para as correções necessárias.

Seria interessante se as etapas da construção do compilador exigissem módulos menores, e um maior número de entregas, e uma gramática mais simples, para facilitar o entendimento dos conceitos e o isolamento dos erros ao longo do processo de desenvolvimento.